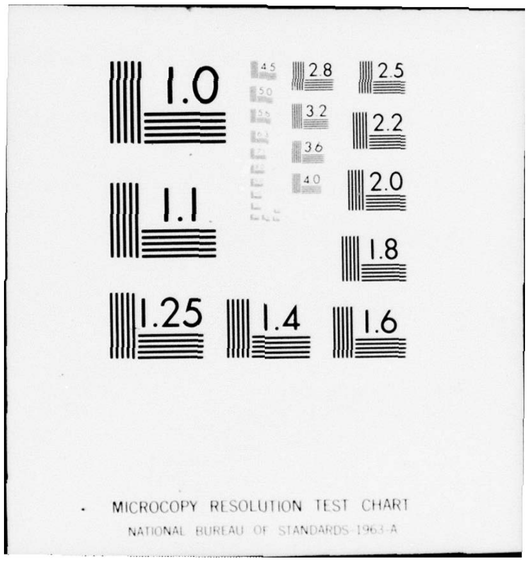
AD-A042 916 IBM THOMAS J WATSON RESEARCH CENTER YORKTOWN HEIGHTS N Y F/G 12/1
AN ALGORITHM FOR MINIMIZING A DIFFERENTIABLE FUNCTION SUBJECT T--ETC(U)
MAR 77 R K BRAYTON, J CULLUM F44620-76-C-0022
UNCLASSIFIED RC-6429 AFOSR-TR-77-0904 NL

| OF |
ADA042916

END
DATE
FILMED
9 -77
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

RC 6429 (#27722) 3/10/77
Mathematics 48 pages

3

Research Report

AN ALGORITHM FOR MINIMIZING A DIFFERENTIABLE FUNCTION SUBJECT TO BOX CONSTRAINTS AND ERRORS‡

R. K. Brayton and Jane Cullum*
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Typed by Candi Brown on CMC (RKB.964)

[illegible]

IBM Research Division
San Jose · Yorktown · Zurich

RC 6429 (#27722) 3/10/77
Mathematics 48 pages

AN ALGORITHM FOR MINIMIZING A DIFFERENTIABLE FUNCTION SUBJECT TO BOX CONSTRAINTS AND ERRORS‡

R. K. Brayton and Jane Cullum*
IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

Typed by Candi Brown on CMC (RKB.964)

ABSTRACT

We consider the problem of minimizing a differentiable function of n parameters with upper and lower bounds on the parameters. The motivation for this work comes from the optimization of the design of transient electrical circuits. In such optimizations the parameters are circuit elements, the bound constraints keep these parameters physically meaningful, and both the function and the gradient evaluations contain errors. We describe a quasi-Newton algorithm for such problems. This algorithm handles the box constraints directly and approximates the given function locally by nonsingular quadratic functions. Numerical tests indicate that the algorithm can tolerate the errors, if the errors in the function and gradient are of the same relative size.

‡ This paper was presented at the SIAM National Meeting, June 16-18, 1976, Chicago, Illinois.

* This Research was sponsored in part by the Air Force Office of Scientific Research (AFSC), United States Air Force, under Contract F44620-76-C-0022.

Copies may be requested from:
IBM Thomas J. Watson Research Center
Post Office Box 218
Yorktown Heights, New York 10598

Introduction

We consider the following optimization problem: Given a real-valued differentiable function f of n parameters $x = (x_1,\ldots,x_n)$, find an xopt such that $f(xopt) \leq f(x)$ for all x in the box $C = \{x \mid a_j \leq x_j \leq b_j,\ 1 \leq j \leq n\}$. Lower bounds $a_j = -\infty$, and upper bounds $b_j = +\infty$ are allowed. We will describe an efficient variable metric algorithm designed for this class of problems where f and its gradient may be evaluated with errors. At each iterate x, we determine a direction of movement d. Then a scalar α^* is chosen such that the next iterate is $x^* = x + \alpha^* d$ and $f(x^*) \leq f(x)$. The change in x and in the gradient g are used to update an approximation H to the inverse of the Hessian of f. Typically, d is the corresponding approximation to the Newton direction on some subset of the constraints at x.

We note that "box" constraints can be used to restrict the parameters to a region where f is well-defined or where the parameters are physically meaningful. Box constrained problems arise naturally in circuit design problems which served as the primary motivation for this work (See section 2).

Members of the one parameter variable metric family, Broyden [2],

$$H_\phi = H - \frac{(Hy)(Hy)^T}{a} + \frac{ss^T}{b} + \phi a \left(\frac{s}{b} - \frac{Hy}{a}\right)\left(\frac{s}{b} - \frac{Hy}{a}\right)^T \qquad (1)$$

have been used to generate quasi-Newton algorithms of the above type for minimization with and without constraints. Fletcher [3] and Powell [4] survey much of the relevant literature. Gill and Murray [5] contains additional information and bibliographies.

In formula (1) and in the remainder of the paper, s denotes the change in parameters x achieved at the current iteration, y denotes the corresponding change in gradient, g denotes the gradient of f at the current iterate x, H denotes the approximation to the inverse of the

Hessian matrix at x and $G=H^{-1}$. In (1), $a=y^THy$ and $b=y^Ts$. The superscript * denotes updated versions of x, g, s, y, H and G. Q will be used to denote the Hessian of f at the iterate x. $A>o$ will denote a positive definite matrix A.

We also use formula (1). We want to choose an update that can determine the true character of the Hessian matrices (we will not force these approximations to be positive definite), and that can accept arbitrary directions of movement without requiring projections of the Hessian matrix approximations. If the Hessian approximation is indefinite on some iteration, we may want to search along a direction that is not necessarily a quasi-Newton direction on some subset of the constraints. Moreover, if no information is lost when constraints are encountered, we can use the approximate Hessians to tell us, at each iteration, what constraints we should try to drop. These requirements lead us to the update in (1) corresponding to $\phi = b/(b-a)$, known as the symmetric rank one (SR1) update,

$$H^* = H - \frac{(Hy-s)(Hy-s)^T}{y^T(Hy-s)} . \tag{2}$$

We also have an additional requirement that the procedure we choose should not be too sensitive to reasonable errors in the function and gradient evaluations in the sense that it should achieve a minimum to within the specified error. Because the update in (2) can accept arbitrary directions of movement and inaccurate line searches, we therefore expect it to be fairly insensitive to reasonable errors.

To avoid getting lost in detail we will describe only the basic ideas in the algorithm. Where possible, the procedures used are supported by relevant Lemmas and Theorems. Otherwise we try to provide the heuristic reasoning behind the adoption of the procedures.

In section 2 we describe the particular application, electrical circuit design problems, which motivated the development of this algorithm. Typically, for such problems each function and gradient evaluation is expensive, costing many times the cost of the auxiliary variable metric computations. Furthermore, these values are computed with error.

In Section 3, the basic algorithm is outlined. In Section 4, we further motivate our choice of the SR1 update, discuss the difficulties in updating occasionally encountered, and procedures for dealing with such difficulties when they occur. Some of this material is more fully developed in Cullum and Brayton [1]. The updating tests used are directly related to questions of dependency of the directions of search generated and of the singularity of the approximate Hessian matrices generated. We do not require the approximating matrices to stay positive definite, only nonsingular.

In section 5, we describe the procedure used for determining the direction of movement at each iterate. A quadratic program whose size equals the number of active constraints is solved at each interation. If this fails for some reason, a full-sized QP is solved. This differs from other procedures, (see Mangasarian [6] and Fletcher [7]), which solve a full-sized quadratic program at each iteration. In Section 6, we describe the scaling heuristics used. The scales obtained are used to generate minimal step requirements in each line search, to obtain an initial guess on the Hessian approximation, and in determining when a constraint is on a boundary.

In Section 7, we describe briefly the line search procedures used. Bending, due to McCormick [8], is used to avoid zigzagging. Searches are made along lines not rays. Section 8 contains remarks about the use of non–quasi–Newton directions and the convergence test.

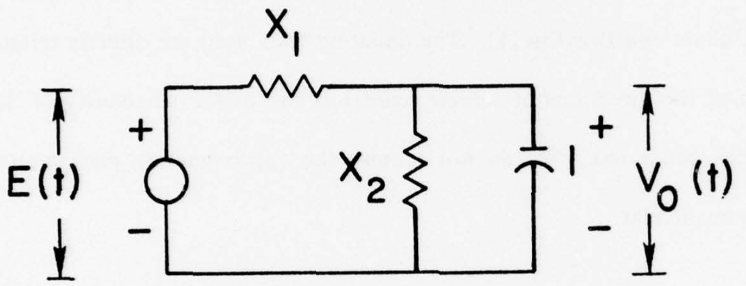
In Section 9, the results of several numerical tests are described. Two types of tests were made. One to demonstrate results for box constraints, and one to demonstrate the behavior of the algorithm when there are errors in the function and gradient evaluations.

Extensions to more general constraints are not discussed.

2. Circuit Design Applications

The following simple example illustrates the use of optimization in time domain, circuit design problems.

Example 1.



Given an input voltage E(t) determine appropriate values of the resistances x_1 and x_2 to make the output voltage $V_o(t)$ match a specified desired voltage V(t) as closely as possible in some norm. For example, minimize, $f(x) = \|V_o - V\|^2_2$ where $\|\cdot\|_2$ is the L_2 - norm. V_o (and thus f(x)) is evaluated by solving a system of ordinary differential equations that describe the voltages and currents in the circuit. In addition, the resistances are constrained in size, $a_j \leq x_j \leq b_j$, i=1,2. For example $x_j \geq o$ is a restriction that if violated would produce a circuit that would not make sense physically.

In general in a transient circuit design problem, the function to be minimized has the form

$$f(x) = \int_o^T h(u(x,t),x,t)dt \qquad (3)$$

with the constraint set $C=\{x \mid a \le x \le b\}$. $u(x,t)$ is the solution of a system of ordinary differential equations which depends on the parameters x:

$$\dot{u} = W(u,x,t) \qquad t \in [o,T]. \tag{4}$$

Using variational arguments (see for example, Hestenes [9]), we can show that the gradient of such a function, $g(x)$, is expressible as the integral of the Lagrange multipliers $\lambda(t)$ associated with the differential equation constraints (4). For example,

$$g(x) = -\lambda^T(0)\frac{\partial u_o}{\partial x} + \int_o^T \left[\frac{\partial h}{\partial x} - \lambda^T(t)\frac{\partial W}{\partial x}(u,x,t)\right] dt \tag{5}$$

where

$$u_o = u(o), \dot{\lambda} = -\frac{\partial W^T}{\partial u}\lambda + \frac{\partial h}{\partial u}, \ \lambda(T) = o$$

and ∂ denotes partial derivative. This method of obtaining the gradient is called the adjoint method. A 3 parameter problem described in Brayton, Hachtel, and Gustavson [10] has, 180 nodes, 460 branches, 112 nonlinear elements and 192 differential equations.

Analytical expressions for f and g are not available. Both are computed with truncation error, due to the finite step size used in solving the differential equations, and each evaluation is expensive. We note, however, that once the function is evaluated at some x, then the corresponding gradient evaluation costs approximately another function evaluation. The truncation error varies from iteration to iteration and is controlled roughly by parameters set by the user. The greater the accuracy required, the longer the time required to evaluate f. Thus, the philosophy of optimization is different here than in the typical minimization literature. We do not want and in fact cannot compute arbitrarily accurate minima; we want a "good enough" approximation. This error situation is illustrated in Figure 1 where the error in the function evaluation of a unimodal function of 1 variable is shown. A similar graph could be drawn for the gradient of f. We know f only to within the tube. Any line search to find the minimum of f in Figure 1 should stop within the indicated stopping region, because to within the specified error, all such x are equally good minima of f. If a better approximation to the minimum is required, the user supplied error control parameters must be tightened, but then each function and gradient evaluation becomes more expensive.

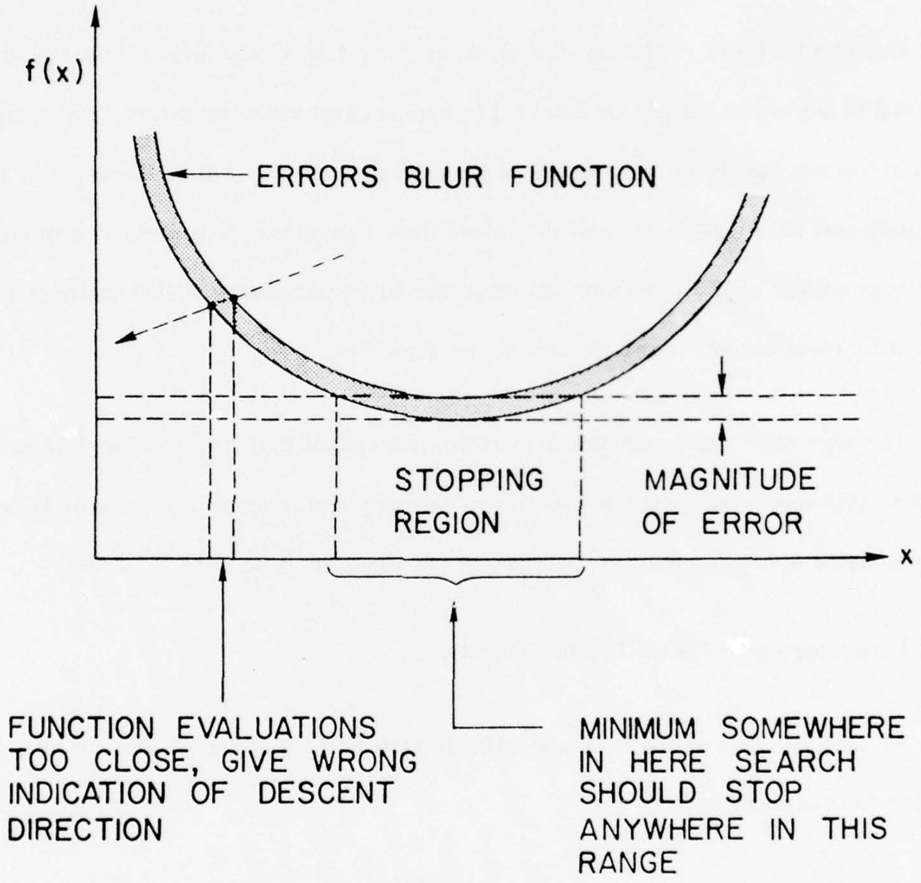
The truncation error affects the line search, the updating, and the convergence tests. Basically, we need a forgiving algorithm, one where the steps do not have to be done exactly.

It had been proposed to one of the authors that the proper approach in such a situation is to consider the output of our function evaluator as an approximation to f, call it f^a, and then to minimize f^a exactly. This makes no sense in this setting, since $f^a=f+e_1$, and the gradient $g^a = g+e_2$, where e_2 is not necessarily related to e_1, and neither e_1 nor e_2 is a well-behaved function. We have to admit that we do not know f and g precisely and must handle the computations accordingly.

The minimization procedure was therefore designed to be flexible and user-controlled. Probably, it is best to use it interactively. Users set parameters determined by their knowledge of the accuracy of f and g. These parameters can be altered to check or improve the accuracy of the solution obtained.

In Section 9 numerical experiments of the following type are discussed. We modify the program that evaluate f and g so that $f_\epsilon = f+\epsilon_1$ and $g_\epsilon = g+\epsilon_2$ are returned to the optimizer where ϵ_1 and ϵ_2 are random variables with $|\epsilon_1| < (a+r|f|)$ and similarly for $|\epsilon_2|$. a represents an absolute error in f and r a relative error.

Figure 1


f(x)
ERRORS BLUR FUNCTION
STOPPING REGION
MAGNITUDE OF ERROR
x
FUNCTION EVALUATIONS TOO CLOSE, GIVE WRONG INDICATION OF DESCENT DIRECTION
MINIMUM SOMEWHERE IN HERE SEARCH SHOULD STOP ANYWHERE IN THIS RANGE

3. Outline of Algorithm.

The algorithm is a quasi-Newton method and assumes that f can be adequately approximated near its minimum by a nonsingular quadratic. As with all quasi-Newton methods, stationary points are computed. There is no guarantee that these are local minimizing points.

Davidon [11] uses the family of updates in (1). Many of the ideas in this algorithm could be used in any extension of Davidon [11] to handle constraints and errors. Although the SR1 update formula has limitations, these can be overcome and this will be discussed in Section 4. We only note here that the tests used to avoid these limitations are connected with maintaining a) independence of the directions searched and b) nonsingularity of the matrices generated. a) and b) are important in any variable metric algorithm.

The algorithm was programmed to store and update both the Hessian (G) and inverse Hessian (H) approximations; however, it can be easily reprogrammed to use only G, and this is demonstrated in later sections as each part of the algorithm is described in detail.

A. Determination of "Best" Feasible Direction.

At each iterate x we have an approximate Hessian G, and we use the quadratic approximation to changes in f at x,

$$\Delta f(x) \backsimeq q(\Delta x) = \frac{\Delta x^T G \Delta x}{2} + g^T \Delta x, \tag{6}$$

to determine a direction of movement at x. If no constraints are active at x, we search along the quasi–Newton line $z = x + \alpha\, Hg$ through x. If one or more constraints are active at x, we solve the quadratic program, minimize $q(\Delta x)$ over all feasible Δx, to obtain the next direction of search. We reduce this to a quadratic program (QP) of size equal to the number of active constraints (see Section 5). If we obtain a solution to this QP which is a direction of descent, and that satisfies other criteria, we search along it. If not, we revert to a full size QP. Note that in using a QP we can generate a direction of movement $d=\Delta x$ that may correspond to dropping no constraints, some, or all of them. Also note that if G is indefinite, such a direction is not necessarily a quasi–Newton direction on some subset of the contraints.

B. Line Search.

We then compute $x^*=x+\alpha^* d$ where α^* is chosen to roughly minimize f on this line. We must search along lines (instead of rays) since G need not be positive definite. Bending, (see McCormick [8],) is allowed in the line search to avoid zigzagging which can occur if we bounce on and off of constraints without achieving minimization. We require that $\|\alpha^* d\|$ be greater than some minimum step because of error considerations (see D and E).

C. Updating.

The resulting overall changes in x, denoted by s, and the changes in gradient g , denoted by y, are used to update G (and H) using the SR1 update formulas:

$$G^* = G + \frac{(y-Gs)(y-Gs)^T}{s^T(y-Gs)}$$

$$H^* = H - \frac{(Hy-s)(Hy-s)^T}{y^T(Hy-s)} \quad . \tag{2'}$$

Updating requires that the denominators in (2′), $s^T(y-Gs)$ and $y^T(Hy-s)$ do not vanish. In Section 4, we discuss the updating tests. If any of these tests fail, the algorithm uses a direction of movement other than a quasi-Newton line on some subset of the constraints. The non quasi-Newton directions discussed theoretically (see Theorems 3 and 4), are eigenvectors of projections of G and the coordinate directions. However, for simplicity, in all the numerical tests run, only coordinate directions were used. Test failures occurred infrequently and were easily handled using coordinate lines.

D. Scaling.

In circuit design applications experience has shown that some advantages can be obtained by scaling the starting Hessian approximation G_0 (and H_0). A heuristic based on the user supplied upper and lower bounds and the initial values of the parameters is used. See Section 6 for details.

E. Error Control.

The user specified tolerances are combined with the scalings generated to obtain tolerances used in (a) determining when a parameter is on a boundary and (b) generating minimal steps used in the line searches. These tolerances can be set to yield a crude approximation to the optimum which can then be refined by reducing the tolerances.

F. Convergence Test.

Since errors are present, the normal convergence tests requiring a small quasi-Newton direction are not appropriate. Therefore, a successive search of the n coordinate lines through a given iterate without achieving descent terminates the procedure. These searches use a minimal step size generated from information provided by the user. We note that with box constraints, at the optimum, the Lagrange multiplier for the j^{th} parameter is simply the j^{th} component of the gradient with the appropriate sign. Therefore, a simple check of the gradients of the active constraints is sufficient to test for stationarity. The procedure also terminates when a consistency test on the function and gradient evauations is violated more than a specified number of times.

4. Symmetric Rank 1 Update.

As stated earlier the symmetric rank 1 update was used because it satifies the requirements listed in the introduction. Theorem 1, see for example Fiacco and McCormick [12], is a statement of most of this fact.

Theorem 1. Let $f=x^TQx/2 + d^Tx$ be a quadratic function. Let s_j, $1 \leq j \leq n$ be arbitrary directions. For any initial x^o, and H^0, successive movement along the s_j with updating on each step using (2), will yield $QH_mg=g$ for some $m \leq n$, if each update is defined.

This theorem can be relaxed to include $k \geq n$ steps, with $H^*=H$ whenever the update is not defined. Murtagh and Sargent [13], [14] and Powell [15] have used the SR1 update in algorithms with mixed results. In Davidon [11] an optimally conditioned update is chosen on each iteration; the SR1 update is the optimally conditioned update in some situations. Powell [16] in analyzing the Davidon [11] work has used an update formula that is a modification of the SR1 update to preserve desirable properties of the Hessian approximations such as positive definiteness. The SR1 update need not stay positive definite, and it is susceptible to ill-conditioning; however, Theorem 1 makes it attractive for constrained problems, where successive directions of movement can be quite general.

The other consideration which lead us to the SR1 update is the following. There are very basic differences between minimization problems with parameter constraints and problems without constraints. For unconstrained problems the assumption, that the function being minimized is locally convex, is necessary for the existence of a minimizer; thus approximation by a positive semidefinite, quadratic function is plausible. However, when constraints are present, local convexity is not necessary for existence, as the simple example $f(x)= -x^2+4, 1 \leq x \leq 2$, demonstrates. We note, however, that it is necessary for the problem to be locally convex in those variables that are interior to the constraints at the minimizer of f. Therefore, since we intend to keep a full Hessian approximation rather than a projected Hessian approximation as is done in Goldfarb [17], we should be able to approximate general quadratic functions not just positive definite ones. With a full approximate Hessian we can utilize the directions of negative curvature to improve convergence. The theory associated with the SR1 update is not restricted to the case where f is a positive definite quadratic function. Therefore, this update can pick up the true character of the Hessian. In regions of negative curvature, the algorithm described here can generate negative curvature directions of search, but does not choose such directions in an optimal way. A drawback, of course, is that the SR1 Hessian approximation can become indefinite even when f is a positive definite quadratic function, so the desire to approximate general functions is only imperfectly satisfied by use of this update. We would really like to have an update formula that could accept general directions of movement and that would simulate the Hessian accurately. The Davidon [11] and Powell [16] analysis may yield such a formula or procedure.

In line with the above comments, and in contrast to most existing algorithms (Murtagh and Sargent [13] and Gill and Murray [18]) we do not try to keep H* positive definite, we worry only about possible dependence of the directions of search generated and singularity of the approximating matrices H* or G*.

Now consider the SR1 update. The denominator in (2′) can vanish if (7.1) $Hy=s$ or (7.2) $Hy\neq s$ but $y^T(Hy-s)=o$. The following theorems tell us the significance of (7.1) and (7.2), and how to circumvent these difficulties.

<u>Theorem</u> <u>2.</u> Consider any variable metric algorithm for which a) at each step H* is given by (1) for some choice of ϕ, b) the direction of search d at each stage is the quasi-Newton direction on some subset of the box constraints, and (c) $x^*=x+\alpha^*d$ where α^* is chosen by some rule. Then;

(a) If f is any function, at some step Hy=s, H is nonsingular, and the active constraint set did not change from x to x^*, then d^* is a multiple of the old direction d.

(b) If f is quadratic, Hy=s at some step, and H is nonsingular, then s is the true Newton direction at x on the current set of constraints.

Note that this result holds for any variable metric algorithm generated using formulas from (1). This theorem, as well as Theorems 3 through 6, was proved in Cullum and Brayton [1] for the unconstrained case.

<u>Proof of Theorem 2.</u>

(a) Assume Hy=s and H nonsingular. Let $\bar{E}$ be the columns of the identity matrix corresponding to the inactive constraints at x. Thus, $\bar{E}^T G \bar{E}$ is the projection of G onto the space spanned by the $\bar{E}$. If we set

$$\Delta f \cong \frac{\Delta x^T G \Delta x}{2} + g^T \Delta x,$$

then the quasi-Newton move s from x to x^* is $s=\alpha\, Ev$ where

$$v=(\bar{E}^T G \bar{E})^{\dagger}\, \bar{E}^T g\ . \tag{8}$$

In (8) $\dagger$ denotes the pseudoinverse of $\bar{E}^T G \bar{E}$. Similarly, the quasi-Newton direction from x^*, if the constraints do not change, is $d^*=\bar{E}(\bar{E}^T G^* \bar{E})^{\dagger}\bar{E}^T g^*$ Since Hy=s, then $g^*=g+Gs$ and $G^*=G$. Therefore, $\bar{E}^T g^* = \bar{E}^T g + \alpha \bar{E}^T G \bar{E} v$ and from (8),

$$\bar{E}^T G \bar{E} v^* = \bar{E}^T g^* = (1+\alpha)\bar{E}^T G \bar{E} v.$$

Hence, v^* differs from v by some vector in the null space of $\bar{E}^T G \bar{E}$. But v^* and v are obtained from the pseudoinverse, which gives vectors whose projections on this null space is zero.

(b) If f is quadratic, then the true Newton direction on the current constraints at x is the vector $\bar{E}w$ where w is the vector of minimum norm that satisfies the equation $\bar{E}^T Q\bar{E}w = -\bar{E}^T g$. Since $Hy=s=\alpha\bar{E}v$, then $Q\bar{E}v = G\bar{E}v$ and $(\bar{E}^T Q\bar{E})v = (\bar{E}^T G\bar{E})v = -\bar{E}^T g$. Therefore, $v=w$.

Q.E.D.

Thus, condition (7.1) indicates dependency problems for the directions of search being generated, if f is not essentially quadratic in the region near x.

Now consider condition (7.2). We have the following theorem, whose proof is independent of the presence of constraints and is given in Cullum and Brayton [1]. It only applies to the SR1 update and connects (7.2) to the singularity of the update G^*.

Theorem 3. For any f, if at some step of an SR1 procedure $Hy\neq s$, but $y^T(Hy-s)=0$, then the SR1 updated Hessian approximation G^*, given by (2′), is singular and $Hy-s$ is a zero eigenvector of G^*. Conversely, if G is nonsingular and $G^*z=0$, then $z = \mu(Hy-s)$. A similar result holds for H^* when $s^T(y-Gs)=o$ but $y-Gs\neq o$.

Theorem 4 tells us how to correct problem (7.1), and Theorem 5 tells us how to handle problem (7.2). Each is an extension of results in Cullum and Brayton [1] to include constraints.

Theorem 4. Let f be any twice continuously differentiable function. Let Q be the Hessian of f at x^*. Let w_k, $1\leq k\leq m$, be orthonormalized eigenvectors of $\bar{E}^T G\bar{E}$ where $\bar{E}$ is the nxm matrix whose columns are + or - the columns of the identity matrix that correspond to the inactive constraints at x^*. Then if the current G does not generate the true Newton direction on the set of constraints at x^*, there exists a w_k such that $(g^*)^T \bar{E}w_k\neq o$, (i.e. $\bar{E}w_k$ is a feasible descent line) and $(HQ-I)\bar{E}w_k\neq o$.

Proof of Theorem 4.

Let

$$W_1 = \{w_k \mid \bar{E}^T G \bar{E} w_k = \bar{E}^T Q \bar{E} w_k\} \text{ and}$$

$$W_2 = \{w_k \text{ not in } W_1\}.$$

If for all $w_k \in W_2$, $w_k^T(\bar{E}^T g^*)=0$, then

$$\bar{E}^T g^* = W_1 v$$

where the columns of W_1 are just the vectors in W_1. Note that each vector in W_1 is an eigenvector of $\bar{E}^T Q \bar{E}$, and therefore, is also an eigenvector of $(\bar{E}^T Q \bar{E})^\dagger$. Therefore, the true Newton direction $\bar{E}z^*$ satisfies

$$z^* = -(\bar{E}^T Q \bar{E})^\dagger W_1 v = -W_1 \Lambda^\dagger v$$

where $\Lambda^\dagger$ denotes the diagonal matrix of the reciprocals of the nonzero eigenvalues of $\bar{E}^T G \bar{E}$ corresponding to W_1 with the reciprocal of any zero eigenvalue set to zero. Similarly, the quasi-Newton direction $\bar{E}z$ satisfies

$$z = -(\bar{E}^T G \bar{E})^\dagger W_1 v = -W_1 \Lambda^\dagger v$$

Therefore, $z^*=z$, contradicting the assumption that the current direction is not the true Newton direction. Thus, there must be at least one $w_k \in W_2$ such that $w_k^T \bar{E}^T g^* \neq o$. But, $w_k \in W_2$ implies that $\bar{E}^T G \bar{E} w_k \neq \bar{E}^T Q \bar{E} w_k$, and since G is nonsingular, $(HQ-I)\bar{E}w_k \neq o$.

Q.E.D.

Thus, theorem 4 says if at some point x^* we have $Hy=s$, then either a) by considering the eigenvectors of $\bar{E}^T G \bar{E}$ we can obtain at x^* a direction of descent $d^*=\bar{E}w_k$ for some k and for small steps α, $Hy^* \neq s^*$, or else b) $-Hg^*$ is the true Newton direction at x^* on the constraint set. Throughout the discussions, if $\bar{E}^T G \bar{E}$ is not of full rank, we call the direction obtained using the pseudoinverse $(\bar{E}^T G \bar{E})^\dagger$ the quasi-Newton direction. This yields the direction of minimal norm satisfying the quasi-Newton stationarity conditions. We should note that since $\bar{E}$ contains only columns of the identity it is trivial to compute $\bar{E}^T G \bar{E}$.

<u>Theorem</u> <u>5.</u> Let $f = x^T Q x/2 + c^T x + e$, where Q is nonsingular. Let $Hy \neq s$, but $y^T(Hy-s)=o$ at some step in an SR1 procedure. Then at x there exists a feasible direction $d=s+\beta e_j$ for some coordinate line e_j and some $\beta \neq 0$ such that for some $\mu \neq o$, $x^{**}=x+\mu(s+\beta e_j)$

is feasible, $f(x^{**}) \leq f(x)$, and

$$(y^{**})^T (Hy^{**}-s^{**}) \neq 0 \tag{9}$$

where $y^{**}=g^{**}-g$, $s^{**}=x^{**}-x$.

Note: For any quadratic, (9) is independent of μ. We write Theorem 5 as above because we must apply it to non–quadratic functions.

<u>Proof of Theorem 5.</u> Define for any e_j and β, $h(\beta)=a_{jj}\beta^2+b_j\beta$ where $a_{jj}= (Qe_j)^T(HQ-I)e_j$ and $b_j= (Qe_j)^T(HQ-I)s$. Then (9) is equivalent to $h(\beta)\neq o$. Since the a_{jj}, $1\leq j\leq n$, are the jth diagonal entries of the indefinite matrix $Q(HQ-I)$, it is possible that all $a_{jj}=o$. However, b_j is the projection of the non-zero vector $(HQ-I)s$ onto Qe_j and Q is nonsingular; hence there must be some e_j such that $b_j\neq o$.

Define $\mathbf{U}_1= \{e_j \mid b_j\neq o \text{ or } a_{jj}\neq o\}$. As in Theorem 4, we let $\bar{E}$ correspond to the inactive constraints at x^*. Note that if $e_j\epsilon\mathbf{U}_1$, then $h(\beta)$ can vanish for at most one nonzero value of β. Moreover, if $h(\beta)=o$ then $h(\beta/2)\neq o$. There are 2 cases to consider.

First assume that there exists an $e_j\epsilon\mathbf{U}_1$ such that $e_j{}^T\bar{E}\ \bar{E}^Tg^*\neq o$. That is, e_j yields a feasible descent line at x^*. In this case we simply search along $y= x^*+\beta e_j$ for a minimum or until we reach an additional constraint. This defines β^*, and if $h(\beta^*)=o$, then β^* is replaced by $\beta^*/2$. Thus, $x^{**}= x+s+\beta^*e_j$ satisfies the requirements of the theorem.

In the second case no such e_j exists. We show that any e_j from $\mathbf{U}_1$ is acceptable. Again there are 2 possibilities. First suppose there is an $e_j\epsilon\mathbf{U}_1$ such that $\bar{E}e_j\neq o$. Then we may use this e_j with β free, but its sign chosen so that $\beta e_j{}^Tg < o$. Since s is a descent direction at x, $s+\beta e_j$ will be also. Then we search for μ so that f is minimized in the direction $s+\beta e_j$ or a new constraint is encountered. The new point is $x^{**}= x+\mu(s+\beta e_j)$. If there is no $e_j\epsilon\mathbf{U}_1$ such that $\bar{E}e_j\neq o$, then all $e_j\epsilon\mathbf{U}_1$ correspond to active constraints. Pick any $e_j\epsilon\mathbf{U}_1$ and choose the sign of β such that, $x+\beta e_j$ is feasible at x for all β of this sign. For example suppose $\beta > o$. $y=x+\beta e_j$ may or may not be a descent ray, but for β small enough $y=x+s+\beta e_j$ is a descent ray. For example, choose

$$|\beta| < \frac{1}{2} \frac{|g^T s|}{|g_j|} .$$

Now we minimize along this ray or move until a new constraint is met. The new point is $x^{**} = x+\mu(s+\beta e_j)$.

Q.E.D.

Observe that the form of s is not used in the proofs of Theorem 3, 4, or 5, so the results are valid for any s, and even when we allow bending. In Theorem 2, s is required to be a quasi–Newton direction on some subset of the constraints. Theorems 2 and 3 generate the tests used on each iteration of our algorithm before updating. Theorems 4 and 5 tell us what to do when one of these tests fails. The tests are as follows.

Test for dependence of directions of search. We check whether

$$\frac{|| \, Hy\text{-}s \, ||}{|| s ||} > \epsilon \tag{10.1}$$

and

$$\frac{|| \, y\text{-}Gs \, ||}{|| y ||} > \epsilon \; . \tag{10.2}$$

For a quadratic, (10) can be written as

$$\frac{|| \, (HQ\text{-}I)s \, ||}{|| s ||} > \epsilon \tag{11.1}$$

and

$$\frac{||(I-GQ^{-1})y||}{||y||} > \epsilon, \tag{11.2}$$

so using (10) we are checking whether s is a zero eigenvector of (HQ-I) and whether y is a zero eigenvector of the counterpart matrix $(I-GQ^{-1})$.

Test for Singularity. Using Theorem 3 we check the following, rather than the denominator of the SR1 update,

$$\frac{||G^*(Hy-s)||}{||Hy-s||} > \epsilon \tag{12.1}$$

and

$$\frac{||H^*(y-Gs)||}{||y-Gs||} > \epsilon. \tag{12.2}$$

We note that we do not have to compute G* or H* to compute these quantities. In fact condition (12.1) equals

$$\frac{|y^T(Hy-s)|}{||Hy-s||}\frac{||y-Gs||}{|s^T(y-Gs)|} > \epsilon \tag{13}$$

and condition (12.2) is just the reciprocal of (13). By checking both G* and H* for singularity, we are (at least heuristically), controlling the condition of each of these matrices by controlling the largest and the smallest eigenvalues of each.

In Cullum and Brayton [1], we give an example using the tests in (12). These tests are violated only infrequently. In fact in the algorithm as programmed, eigenvectors of H are not computed, only coordinates lines are used. In each case that either test (10) or (12) has failed, the simple procedure of taking an e_j that is a direction of descent at x* has given a good update. Typically, only 1 extra line search was needed to complete the update. Theoretically, the procedures proposed could be costly, involving several line searches. However, in practice the cost seems nominal.

If tests (10) and (12) are both passed, then the G and H matrices are updated. Before proceeding we note that only G is required for these tests, since we can compute the vector Hy-s by solving the equation Gp= y-Gs for p.

5. Constrained Directions of Search.

Many linearly constrained variable metric algorithms require minimization on the current set of constraints before any of these constraints can be dropped, (see for example Gill and Murray [18] and Goldfarb [17].) In addition these algorithms often drop only one constraint at a time. Algorithms which allow dropping several constraints simultaneously, generally solve full dimensional quadratic programs at each iteration to determine which constraints to drop, (see for example Fletcher [7] and Mangasarian [6]). For additional comments see Gill and Murray [5].

The scheme proposed below is equivalent to solving a dual to the original problem. This dual is the same size as the number of active constraints. If at a given iteration there are active constraints, we consider the following quadratic programming problem; minimize,

$$q(\Delta x) = \frac{\Delta x^T G \Delta x}{2} + g^T \Delta x \tag{14}$$

subject to $E^T{}_s \Delta x \leq o$. In (14) G is the current Hessian approximation, E_s is the nxm matrix whose columns are the inner normals to the current set of active constraints. Thus q is an approximation to the change in f at x for any admissible Δx. Nominally, this is a full dimensional problem. However, Lemma 1 states that we can replace (14) by a problem whose size equals the number of active constraints. The quadratic programming algorithm in Canon, Cullum and Polak [19] is used to solve this equivalent problem, see (15). If this algorithm finds a solution to (15) such that this Δx is a direction of descent for f, $\Delta x^T G \Delta x > o$, and the diagonal elements of the projection of G onto the free constraints are all positive, then we use this as our direction of search. Otherwise we revert to solving the full–sized quadratic problem in (14), imposing upper and lower bounds and using Fletcher and Jackson [20]. The algorithm in [20] works on any QP; if G is indefinite, then the direction generated need not be a quasi–Newton direction on some subset of the constraints. If the direction generated vanishes, we choose a coordinate line to search. This is equivalent to checking the multipliers and releasing a constraint whose multiplier has the wrong sign.

Lemma 1. The QP in (14) is equivalent to the following QP of size m, the number of constraints active at x: minimize

$$h(v) = \frac{v^T(E_s^T H E_s)v}{2} - v^T E_s^T Hg \tag{15}$$

subject to $v \geq o$. Equivalence means equivalence of the associated necessary conditions of optimality.

Proof of Lemma 1. The necessary conditions of optimality for the problem in (14) are

$$G\Delta x + g - E_s v = o \text{ with } v \geq o,\ v^T(E_s^T \Delta x) = o, \text{ and } E_s^T \Delta x \geq o. \tag{16}$$

Since G is invertible, we have

$$\Delta x + Hg - HE_s v = o. \tag{17}$$

This equation projected onto the active constraints becomes

$$E_s^T \Delta x + E_s^T Hg - E_s^T HE_s v = o,\ v^T(E_s^T \Delta x) = o,\ v \geq o,\ E_s^T \Delta x \geq o. \tag{18}$$

But these are the necessary conditions of optimality for the mxm QP, in (15) where the multipliers for this problem are just the vector $E_s^T \Delta x$. Thus any solution of (16) yields a solution to (18) and conversely, given a solution to (18) we can generate a solution to (16) using the equations in (17) to solve for the unconstrained components. Q.E.D.

If $G > o$, then any solution of (16) yields a minimizing point of q, and Δx is the optimal direction of movement for q.

We note that if only G is available in the program, we can generate the necessary vectors, HE_s, for this computation by solving the equations $GP=E_s$ for the matrix P. We also note that (15) is the dual of (14). Murray [21] mentions using the dual problem, but no details are given.

For this approach to be useful, we need a good quadratic programming algorithm. Obviously, if $E_s{}^T HE_s$ is indefinite, (15) has no solution. If we put an upper bound on Δx, then a solution could exist, but the reduction in dimensionality would not occur. The quadratic programming algorithm described in Canon, Cullum and Polak [19] has the following properties. Consider the QP, minimize

$$\frac{z^T Az}{2} + d^T z$$

subject to $Fx \geq o$. Then (a) if $A \geq o$ and a finite solution exists, this algorithm will find that solution, (b) if $A \geq o$ and there is no finite solution, the procedure will generate an infinite ray, and (c) for general A, in a finite number of steps, this procedure either generates a z that satisfies the stationarity conditions or indicates that no solution exists. For our problem (15), (b) cannot occur.

Lemma 2. If $G \geq o$, then $E_s{}^T HE_s \geq o$ and the QP in (15) always has a finite solution.

Proof of Lemma 2. Since $E_s{}^T HE_s$ is the projection of $H \geq o$ onto the space spanned by E_s, it must be positive semi-definite on this subspace. No infinite solution can exist since the null space of $E_s{}^T HE_s$ and of the constraint $v \geq o$ is trivial.

Q.E.D.

The use of (15) allows us to work with QP's of the same size as the current number of active contraints. Both QP's allow us to drop none, one, some or all of the active constraints. The QP gives us only the direction of search and not the step size which is obtained by a line search.

6. A Scaling Heuristic.

In the circuit design applications, because of wide variations in scale between different elements of the circuit, some advantages can be obtained by scaling the starting Hessian matrix G_o. We want to determine a matrix P such that the function $F(x^1) \equiv f(Px^1)$ is better conditioned than $f(x)$. Powell [22] has shown that by simply setting $G_o = P^{-T}P^{-1}$ instead of I, subsequent steps in the x-space are identical to steps that would be taken in the x^1 space if we explicitly made this substitution and worked directly with $F(x^1)$.

The following heuristic was used to generate a P. We use a simple diagonal scaling. It is assumed that the user has supplied realistic upper and lower bounds a_j and b_j, $1 \le j \le n$, and/or a reasonable initial value x_j^o for each parameter. Then p_{jj} is computed as follows.

$$p_{jj} = \begin{cases} \min(|x_j^o|, (b_j - a_j)) & \text{if } |x_j^o| \ge 1 \text{ and } 1 \le b_j - a_j < \infty \\ \max(|x_j^o|, (b_j - a_j)) & \text{if } |x_j^o| \le 1 \text{ and } b_j - a_j \le 1 \\ \max(1, |x_j^o|) & \text{if } b_j - a_j = \infty \\ 1 & \text{otherwise} \end{cases}$$

For example if $x_j^o = 10^{-2}$, $b_j - a_j = 10^{-4}$, then $p_{jj} = 10^{-2}$. If $x_j^o = 10^2$, $b_j - a_j = 10^4$, then $p_{jj} = 10^2$.

Whenever the algorithm is restarted the G_o matrix is rescaled using the current iterate.

We note that setting G_o equal to a diagonal matrix may mean that $G_n \not\to Q$ in some problems. In particular this may happen if the variables are separable. But, of course, we do not need $G_n \to Q$, we only need QHg to approximate g. We note also that as indicated in Bard [23] scaling or lack of it in any of the variable metric schemes can cause serious distortions in the approximating matrices and lead to near singularities. In the minimization procedure the user should include all the information about the given problem that is available, and should anticipate difficulties. When difficulties occur, one obvious thing to try is rescaling the G matrix, perhaps using the relative sizes of the current gradient components as scalings on each component.

The scale p_{jj} are also used in the program with the user-set tolerance δ for (a) testing when a parameter is on a boundary and (b) generating minimum steps that are used in the line searches. These tolerances can be set to yield a crude approximation to the optimum which can then be refined by reducing the tolerances.

7. Line Search Procedures.

For completeness the line search is outlined, but no claims for the superiority of this method are made. The new elements are the minimal step requirement which is put in to deal with errors, and the use of bending. Except for the gradient at x, the line search uses only function values and thus can be used even if the gradient is being approximated.

The initial step along the line of search is obtained by using the current quadratic approximation G. This first step is either $\|d\| \; |g^t d| / |d^t G d|$ or the distance to the closest boundary, which ever is smaller. We note that for directions of negative curvature this is not an optimal choice. If a constraint is encountered before the minimum is bracketed, we project d onto the new constraint and continue to move along the bent line, until a bracket on the minimum of f is achieved. Once the minimum is bracketed a quadratic is fit to the 3 function values that form the bracket. Steps out are obtained using quadratic extrapolation. If a concave fit is made on some step, (indicating negative curvature), a larger extrapolation outward is made. Observe that if we fit a quadratic on a bent portion of the line of search, see Figure 2, then we can expect only a crude fit, since the one-dimensional quadratic approximation to f would be different on each arm of the bend.

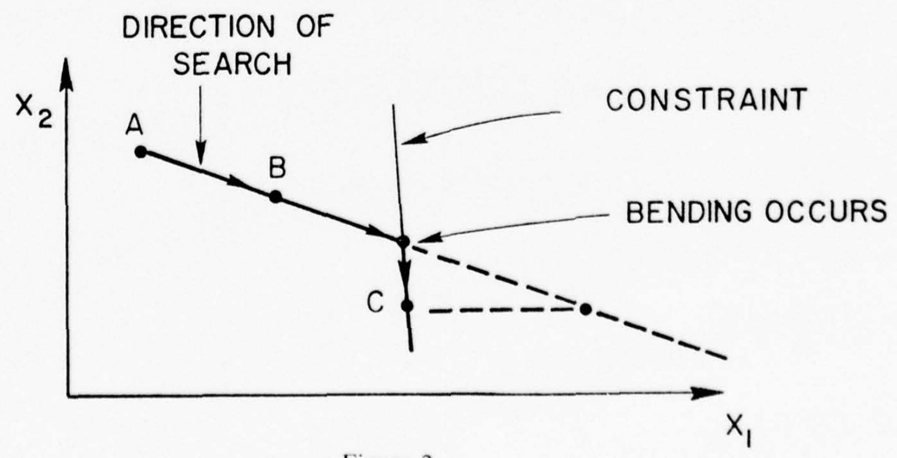


Figure 2

One could avoid this problem by computing gradients at the 3 points in the bracket and then taking the proper combination of gradient and function values at 2 of the points to fit a quadratic on the associated subinterval. We, however, did not do this. In our implementation, the bending is done in the function evaluation subroutine and not in the line search subroutine. The steps in the line search are taken along the original line, but the function is evaluated at the projections of these points onto the constraints, see Figure 3.

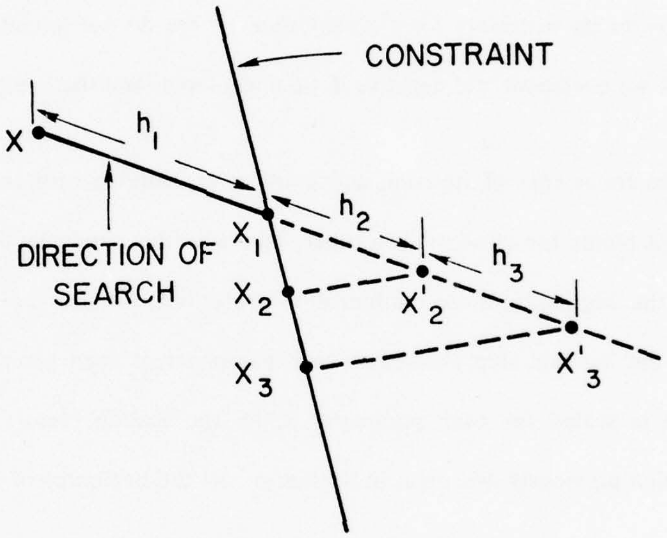


Figure 3

The quadratic fit is made along d using the values x_1, x_2', x_3', however, in each case, with the function evaluated at the projected points x_1, x_2, x_3. This simplifies the search routine. As shown by McCormick [8] bending prevents jamming or zig-zagging, the phenomenon where the minimization procedure is completely controlled by the constraints (Zoutendijk [24]). Note that when bending occurs, the change in x, $s = x^*-x$, is not in the initial direction d, hence is not a quasi–Newton direction.

Even though we are using the SR1 update we require line searches because they are beneficial in controlling the condition of the matrices generated. If f is a quadratic function, then each successful update increases the dimension of the space S, where $s \epsilon S$ implies that $Gs=Qs$. Obviously, if the directions s are arbitrary, then the overall problem, find Q from $QS=Z=GS$ can be arbitrarily poorly conditioned. However, with line searches, the successive directions are more likely to be nearly conjugate and thus sufficiently independent.

Minimization also helps maintain positive definiteness. If $G>o$ and $s^T(y-Gs)>o$, then $G^*>o$, although this is not necessary in order that $G^*>o$. If we minimize, then $s^Ty \approx -g^Ts>o$ which is certainly necessary for $s^Ty>s^TGs>o$. If we do not *minimize, then* $(g^*)^Ts$ *would be* positive if we overshoot and negative if we undershoot, and there is no guarantee that $s^Ty>o$.

In the framework of function and gradient evaluations with error, it is important not to evaluate at points too close to each other, otherwise the error may dominate the minimization. Prior to the beginning of the optimization procedure, the user has to specify a tolerance δ, *which is the minimal step* allowed in each parameter, if each parameter is scaled to 1. This tolerance is scaled for each parameter x_j by the scaling factors p_{jj} computed in the H^o computation previously described in Section 6. At the beginning of each line search a minimal step

$$\min_{1 \le j \le n} \left| \frac{\delta p_{jj}}{d_j} \right| \tag{20}$$

is computed, where d is the direction of search. For example, if $10^{-4} \le x_1 \le 10^{-2}$, $10 \le x_2 \le 100$, and $x^o = (10^{-3}, 20)$, then $p_{11} = 10^{-2}$ and $p_{22} = 20$. Then if the function had been evaluated at x no further function evaluation is allowed in the set $\{x \mid |x_1 - x_1| \le 10^{-2}\delta, |x_2 - x_2| \le 20\delta\}$. Thus, it is possible, even though $g^Td < o$, for the search to indicate that no descent was achieved, if a step larger than the minimal step could not be taken. In this case an alternative line is searched (as implemented a coordinate line). Probably a better choice would be an eigenvector of G. Ideally, the tolerance δ is related to the accuracy achievable in f and g.

Two line searches were considered, one attempting to bracket the minimum on a line, the other using a ratio of the reduction in f last achieved on that line to the estimated next reduction achievable. Numerical results for both are given in Section 9 as search 1 and search 2, respectively. Not requiring bracketing typically reduces the number of function evaluations required, but increases the number of iterations. Thus, one would expect search 1 to be best for problems where gradient evaluations are significantly more expensive than function evaluations, and search 2 to be best for problems where these costs are approximately equal.

In the next section we describe the convergence test used and motivate the use of other directions of movement (e.g. coordinate lines or eigenvectors) in addition to the quasi-Newton directions.

It is desirable for all variable metric algorithms to have a restart mechanism. In our algorithm the following heuristic is used for restarting. *It is tied to the line search. If (a) $n+1$* updates have been achieved since the most recent restart, and (b) on each of 3 successive iterations, 5 or more function evaluations are required in the line search, then the procedure is restarted. H_0 is rescaled on any restart. 3 and 5 were chosen heuristically. We probably should have excluded directions of negative curvature from this test; however, this was not done. In the tests run, restarts occurred only infrequently. The idea of throwing away all the information accumulated to date and starting fresh does not appeal to us.

8. Alternative Directions of Search and the Convergences Tests.

In addition to the non–quasi–Newton directions that can be generated by the full QP, two other families of vectors can be used when the quasi-Newton directions on subsets of the constraints are not acceptable. Whenever $Hy=s$, Theorem 4 suggests that we use an eigenvector w_k of $\bar{E}^T G \bar{E}$ such that $s=\alpha \bar{E} w_k$, $Hy \neq s$ and $(g^*)^T s \neq o$. Theorem 5 says that if $Hy \neq s$, but $y^T(Hy-s)=o$, then we can find a suitable coordinate vector e_j which will allow an update. In the numerical results described the eigenvectors of $\bar{E}^T G \bar{E}$ were not computed, coordinate lines were used in both situations. Initially the coordinate lines were ordered according to the size of g_j, scaled using the p_{jj} generated in finding H^o. The coordinate lines were considered cyclically within this ordering. Coordinate lines were also used whenever the inner product of the projected gradient with the computed direction of movement vanished. Since we do not require H and G to be positive definite, $w^T H w$ can vanish when $Hw \neq o$.

If a successive check of all the coordinate lines at a given iterate yields no descent, then the procedure terminates. When we stop, because of the minimum step requirement, we are only near a Kuhn-Tucker (hopefully minimum) point. The distance from the 'minimum' depends upon the distortion of the local contours of the function being minimized. The following Theorem gives an estimate of this error in the unconstrained case for f a quadratic function.

<u>Theorem 6.</u> Let f be a quadratic function, $f(x)= x^T Q x/2 + d^T x$ where $Q>o$. For any $\delta>o$ and scales p_{jj}, $1 \leq j \leq n$, if at some point x^*, $f(x^* \pm \delta p_{jj} e_j) > f(x^*)$ for all $1 \leq j \leq n$.

Then

$$|g_j| < p_{jj} \delta Q_{jj}/2$$

and

$$||x^* - xopt|| \leq \frac{\delta}{2\underline{\lambda}(Q)} (\Sigma p_{jj}{}^2 Q_{jj}{}^2)^{1/2}$$

where $\underline{\lambda}$ (Q) is the minimum eigenvalue of Q.

Proof. Since f is a convex quadratic function, for each j and for all $|\alpha| > p_{jj}\delta$,

$\phi_j(\alpha) \equiv f(x^* + \alpha e_j) - f(x^*) = \alpha g_j{}^* + \alpha^2 Q_{jj}/2 > o$.

The minimum of $\phi_j(\alpha)$ occurs at $\alpha_j = -g_j{}^*/Q_{jj}$ and $\phi_j(\alpha_j) < o$. Furthermore, $\phi_j(o) = o = \phi(2\alpha_j)$. Therefore, $\phi_j(\alpha) < o$ in the interval $[0, 2\alpha_j]$ and therefore, $p_{jj}\delta > 2\alpha_j$. Substituting for α_j and rearranging we get

$$|g_j{}^*| < p_{jj}\delta Q_{jj}/2. \tag{21}$$

Combining (21) and the fact that $\|gopt\| = o$, we get for each j

$$|[Q(x^* - xopt)]^j| < p_{jj}\delta Q_{jj}/2$$

Therefore,

$$\underline{\lambda}^2 \; ||x^* - xopt||^2 \leq \; ||Q(x^* - xopt)||^2 \leq \frac{\delta^2}{4} \Sigma p_{jj}{}^2 Q_{jj}{}^2$$

or

$$||x^* - xopt|| \leq \frac{\delta}{2\underline{\lambda}(Q)} (\Sigma p_{jj}{}^2 Q_{jj}{}^2)^{1/2},$$

where $\underline{\lambda}$ (Q) is the smallest eigenvalue of Q.

Q.E.D.

Theorem 7. If we used the eigenvectors of Q, v_k, $1 \leq k \leq n$, in Theorem 6, then we obtain

$$|v_j{}^T g^*| < \frac{\lambda_j \delta}{2} [\min_k |p_{kk}/{v_j}^k|] \tag{22}$$

and

$$||x^*-xopt|| < \frac{\delta}{2} \left(\Sigma[\min_k |p_{kk}/_{v_j}k|]^2 \right)^{1/2} \qquad (23)$$

In (22) and (23) the minimum is taken over those k with $v_j^k \neq o$.

<u>Proof.</u> For the eigenvectors $v_j, 1 \leq j \leq n$, the minimal allowed step in the line search along v_j corresponding to the scales $p_{kk}, 1 \leq k \leq n$, is $\delta \min_k(p_{kk}/v_j^k)$. The revelant expansion is $\phi_j(\alpha) = f(x^*+\alpha v_j) - f(x^*) = \alpha g^* \nu_j + \alpha^2 \lambda_j/2$ where λ_j is the eigenvalue of Q corresponding to v_j. But the minimum of ϕ_j occurs at $\alpha_j = -g^* v_j/\lambda_j$, $\phi_j(\alpha_j) < o$, $\phi_j(o) = o = \phi_j(2\alpha_j)$. Therefore,

$$\delta \min_k (p_{kk}/_{v_j}k) > 2\alpha_j .$$

Rearranging we obtain (22).

Since f is quadratic, $Q(x^*-xopt) = g^*$. Let V be the matrix whose columns are the v_j, $1 \leq j \leq n$. Then introducing V as a basis, we obtain

$$(V^T Q V)\, V^T(x^*-xopt) = V^T V (V^T g^*)$$

or

$$\Lambda V^T(x^*-xopt) = V^T g^*$$

where Λ is the diagonal matrix with entries λ_j, $1 \leq j \leq n$. Therefore, from (22)

$$|v_j^T(x^*-xopt)| < \frac{\delta}{2} [\min_k (p_{kk}/v_j^k)]$$

Summing over components gives us (23).

Q.E.D.

The absence of $\underline{\lambda}(Q)$ in the estimate (23) tells us (as expected) that the directions v_j are a better choice than the e_j. Thus, if G is a good approximation to Q, we would probably do better to use the eigenvectors of G rather than the coordinate lines.

In the next section we present some numerical tests, problems with and without box constraints, and with and without noise added to the function and gradient evaluations.

9. Numerical Results

A computer program was developed as the optimization part of a circuit design package and has been used on circuit design problems. However, it does not seem practical to use such functions as test problems since other researchers would not have access to the circuit generating package used to evaluate these functions and gradients. Therefore, so that future comparisons can be made, we present some results for a few other functions; most are standard test functions found in the optimization literature. The test functions used are listed below with their starting values, and bounds.

Constrained Problems

1) FUNCTION FHOLZ (Himmelblau [25])

$$f(x_1,x_2,x_3) = \sum_{i=1}^{99} \left(e\left(\frac{-(u_i-x_2)^{x_3}}{x_1}\right) \quad -.01i\right)^2$$

$$u_i = 25 + (-50 \ln(.01i))^{\frac{1}{1.5}}$$

Bounds: $.1 \le x_1 \le 100$

$0 \le x_2 \le 25.6$

$0 \le x_3 \le 5$

$x^o = (10, 1.25, .3)$

$xopt = (50, 25.0, 1.5)$

$f(xopt) = 0.$

2) FUNCTION FRECP (Dixon [26])

$$f(x_1,x_2,x_3) = (x_2-5)^2 + (x_1+x_2^2)^2 + x_3^2/x_1$$

Bounds: $10^{-3} \leq x_1$

$x^o = (1,2,1)$

$xopt = (10^{-3}, 1.234, 0)$

$f(xopt) = 16.5045855$

3) <u>FUNCTION</u> <u>FB3</u>

$$fB3(z_1,z_2,z_3) = f(100z_1, .01\, z_2, z_3)$$

where $f(x,y,z)=\alpha z^2(1-\beta((x+y-3)^4 + (y-x-.8)^4))$

$+\gamma((x+y-3)^2 + (y-x-.8)^2)$

$+2(1-z)^5(-9.95x^2 + 1.25xy + 2.48y^2)$

$\alpha = .001$, $\beta = 2500$, $\gamma = 5$

Bounds: $.01 \leq z_1 \leq .02$

$100 \leq z_2 \leq 200$

$0 \leq z_3 \leq 1$

$z^o = (.019, 110, .9)$

$zopt = (.02, 110.29 , 0)$

$fB3(zopt) = -53.5985294$

4) <u>FUNCTION</u> <u>FB6</u>

$$fB6\ (z_1,z_2,z_3,z_4,z_5,z_6,) = fB3(z_1,z_2,z_3)\ fB3(z_4,z_5,z_6)$$

Bounds: $.01 \leq z_1,z_4 \leq .02$

$100 \leq z_2,z_5 \leq 200$

$0 \leq z_3, z_6 \leq 1$

$z^o = (.0151, 151, .921, .015, 150, .92)$

zopt : there are several minima,

$f(zopt) = -275.49644, -346.091, -402.311332$

<u>Unconstrained Problems</u>

(1) <u>FUNCTION FROSE</u>

$$f(x_1,x_2) = 100\,(x_1^2 - x_2)^2 + (1-x_1)^2$$

$x^o = (-1.2, 1.0)$

$xopt = (1,1)$

$f(xopt) = 0.$

(2) <u>FUNCTION FWOOD</u>

$$f(x_1,x_2,x_3,x_4) = 100(x_2-x_1^2)^2 + (1-x_1)^2 + 90(x_4-x_3^2)^2$$
$$+ (1-x_3)^2 + 10.1((x_2-1)^2 + (x_4-1)^2) + 19.8(x_2-1)(x_4-1)$$

$x^o = (-3,-1,-3,-1)$

$xopt = (1,1,1,1)$

$f(xopt) = 0.$

(3) <u>FUNCTION FEASY</u>

$f(x) = x^t Qx - x^t g$

where $Q = \begin{pmatrix} 96.45 & 53.23 & 78.98 & 61.33 \\ 53.23 & 45.93 & 62.14 & 45.11 \\ 78.98 & 62.14 & 89.14 & 62.45 \\ 61.33 & 45.11 & 62.45 & 47.05 \end{pmatrix}$

and $g = (1, 4, 2, 3)$

$x^o = (0, 0, 0, 0)$

$xopt = (.1304 , .8245 , -.4068 , -.3886)$

$f(xopt) = -11.7182934$

(4) <u>FUNCTION</u> <u>FPOWL</u>

$$f(x_1,x_2,x_3,x_4) = (x_1 + 10\, x_2)^2 + 5\, (x_3-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4$$

$x^o = (3, -1,0,1)$

$xopt = (0,0,0,0)$

$f(xopt) = 0$

Of the constrained problems, FHOLZ (Himmelblau [25]) and FRECP (Dixon [26]) have appeared in the literature. FB3 and FB6 were constructed specifically for these tests as problems that would involve much hitting and releasing of constraints before a local minimum could be achieved; and moreover, so that their Hessians are indefinite at each of the minima.

We note that the initial point used in the FHOLZ tests was not the same as that used in Himmelblau [25]. At the starting point in [25] FHOLZ is very flat and the 2 Harwell methods used would not move from that point. To obtain comparisons, we therefore, moved to a more reasonable starting point. The starting points for FB3 and FB6 were chosen such that the procedure must circumvent a saddle point to get to a minima.

FRECP as stated in Dixon [26] involves a constraint $y_2 > y_1^2$. We have transformed the variables and mapped this constraint into $x_1 > 0$. Of course at $x_1 = 0$, FRECP is not differentiable; near $x_1 = 0$ it has steep derivatives however, the constraint $x_1 \geq 10^{-3}$, is not too severe, and all the algorithms behave well.

For the unconstrained problems, FROSE,FWOOD, and FPOWL are standard test functions. FROSE and FWOOD each have a banana–shaped valley, FWOOD also has regions of negative curvature and saddle points. FPOWL has a singular Hessian at its optimum. FEASY is a simple positive definite quadratic function; everything should work well on it.

For the purposes of comparsion we compared the new algorithm(s) with the corresponding algorithms in the Harwell Subroutine Library [27]. Those used were

<u>Harwell name</u>	<u>Referred to as</u>
VE03A	CFletcher
VE05AD	Buckley
VA09AD	Fletcher
VA04AD	Pow–no–drv

The first two, VE03A and VE05AD, are variable metric algorithms for linearly constrained problems with box constraints. VE03A is due to Fletcher [28] and solves a full quadratic programming problem in a variable size box at each iteration. VE05AD is due to Buckley [29] and is a modification of Goldfarb's linearly constrained algorithm [17]. The last two, VA09AD and VA04AD, are for unconstrained problems. VA09AD is a variable metric algorithm due to Fletcher [30]. VA04AD is Powell's no–derivative procedure [31]. VE03A was only available in the single precision version; roundoff in the 370–168 could be 10^{-6}, so the results for CFletcher should be viewed in this light.

Tests were run on the eight test functions with and without noise. For each function, noise, e, in the function values was simulated by specifying a relative error, r and an absolute error a, and then setting the error,

$$e = \gamma\,(r\,|f|\ + a)$$

where γ is a uniformly distributed random variable in the interval $[-1,1]$. Noise in the gradient was generated similarly. We realize that this imperfectly simulates the effects of truncation error for functions that depend on the solutions of differential equations. But, we do not have a good model for the truncation error, and we expect that the simulated effect is worse than the truncation effect. Thus, if a procedure works reliably in the presence of our simulated error we expect it would work well in the presence of truncation error.

The results of all tests are in Table 1. They are separated by function, each has a run with no noise, with single precision noise, and with a lot of noise. The number of functions and gradients required for each procedure is given. For each of the Harwell routines, the convergence criterion was $|\Delta x_i| \leq 10^{-4}$, $1 \leq i \leq n$. For the algorithm described in this paper, the tolerance $\delta = 10^{-4}$, and the minimum Δx was obtained by scaling as described in Section 7. Convergence occurred when a successive search at minimum Δx of the coordinate lines through the current iterate yielded no descent; or a consistency check, designed to spot when the errors in the gradient or function evaluations are dominating the minimization had been violated a total of 3 times.

In the test results in Table 1, the algorithm described in this paper is called BCR1, for $\underline{b}$ox $\underline{c}$onstrained $\underline{r}$ank $\underline{1}$. Four versions of BCR1 were tested. BCR1(1,1) and BCR1(2,1) are direct implementations of the procedures described, differing only in the line search used. Line search 1 requires more function evaluations and attempts to bracket the minimum before leaving the search. Line search 2 leaves the search as soon as it has achieved a reasonable decrease in the function. Both searches use only function values with the exception of the gradient at the starting point of the search. Use of line search 2 often results in an increase in the number of iterations over line search 1 and thus an increase in the number of gradient evaluations. In the circuit design applications, once a function evaluation is made, a gradient evaluation costs the same as another function evaluation, so search 2 is reasonable. However, in other applications gradient evaluations may cost some multiple of a function evaluation. For example, if we are computing derivatives by differencing, we would want to minimize the number of iterations and hence line search 1 would be more appropriate.

BCR1(1,1) and BCR1(2,1) both solve a quadratic subprogram only on those iterations where at least one constraint is active. Each first attempts to solve a small QP using the QP algorithm in Canon, Cullum and Polak [19]. If this fails (which can happen only if G is indefinite), or if $d^tGd < 0$, $g^td > 0$ or $\bar{E}^tG\bar{E}$ has a negative diagonal element where $\bar{E}$ is that part of the identity corresponding to the variables designated by the small QP as free, then a full-sized QP (using the bounds min $(|x_i-t_i|, 100p_{ii})$, $1 \leq i \leq n$, where t_i is the appropriate upper or lower bound) is solved using Fletcher and Jackson [20].

A modified version of BCR1 was also tested, and this is labelled BCR1(1,4) and BCR1(2,4) in the table where 1 and 2 again correspond to the 2 searches used. BCR1(•,4) solves the full-sized QP at each iteration to determine the direction of search. One might expect, since this approach is global, that convergence should improve over BCR1(1,1) and BCR1(2,1). However, the numerical results indicate that this is not true.

The total work required is given as the sum of FCNS plus GRDS. This only makes sense if the function and gradient evaluations have equal cost. Otherwise, the 2 terms must be weighted by their relative costs. Moreover, we are concerned here with problems where the cost of a function or gradient evaluation dominates the cost of the computations done in the optimization algorithm.

On this basis of work done, we would judge BCR1(2,1) to be the best of the four options for BCR1. This method proved to be reliable and efficient in the presence of box constraints and noise. Furthermore, the results indicate that it compares favorably with VA09AD, an excellent unconstrained method (judged by Himmelblau to be the best of those tested by him). Performance did not degrade in the presence of noise. In all cases BCR1(2,1) terminated at a reasonable point, either at or near a local minimum, or at a point where it could not be expected to proceed because the noise was of the same magnitude as the function.

BCR1(1,1) was as reliable as BCR1(2,1). In most cases, it used more function values but fewer gradients then BCR1(2,1). Thus, one might expect it to be more efficient on problems where the cost of computing a gradient is a multiple of the cost of a function evaluation.

We had expected that BCR1(1,4) and BCR1(2,4) would perform better than BCR1(1,1) and BCR1(2,1) since they use global information on each iteration. However, they did not. Perhaps the size of the box used with the algorithm in [20] was too large or should have been allowed to vary as is done in VE03A (CFletcher). However, the results in Table 1 for CFletcher do not indicate that this would be a profitable direction to pursue; since CFletcher did not perform well on any problem except FRECP. This comment should be tempered with the comment that CFletcher is a single precision routine. As the table indicates for FB3, using the scaling suggested by Fletcher, termination at the saddle point occurred. However, using no scaling, the procedure went to the minimum of FB3. For FB6 it went to the saddle point on all scalings tried. For FHOLZ termination occurred at a noncritical point.

VE05AD (Buckley) did not perform reliably on any function except FRECP. As indicated for FB3 and FB6, when the initial point was shifted closer to a local minimum, VE05AD produced that local minimum. This leads us to believe that we had the procedure set up properly. The output of VE05AD obtained indicates that not enough safeguards are built into the line searches. In particular, on some runs bounds were violated, and when noise was present the procedure did not recognize it and terminate the iterations when the noise dominated the minimization. On FB3 and FB6, it terminated normally, but at points that are not local minima or saddle points. In each of these 2 cases it failed to allow movement along a good coordinate direction. As stated earlier FB3 and FB6 were designed to be difficult to minimize because they require much hitting and releasing of constraints. Moreover, the Hessians are indefinite at the minima.

For the unconstrained algorithms, VA09AD (Fletcher) exhibited good behavior in the presence of noise on all functions except FWOOD. We note that with a smaller convergence criterion Fletcher did achieve the minima of FWOOD when there was no noise. Remarks in the Harwell write-up of this program indicate that some thought was given to its performance in the presence of errors and certainly its performance in the presence of error substantiates this.

Tests on Powell's no derivative algorithm are included to see if a procedure that uses only function values but does not use differencing to get derivatives, and that works on the principle of conjugate directions, would work well in the presence of noise. Indications are that it works well with a moderate amount of noise, at least on the small dimensional problems tested. With a lot of noise, it did not work as well as the variable metric algorithms.

The work-counts for the Harwell routines were obtained as follows. Both Cfletcher and fletcher have a function, gradient subroutine that evaluates both simultaneously, so that in Table 1, in all cases for these two methods FCNS=GRDS. From the write-ups, it was not clear whether each time the function is called, the gradient information is also used in these algorithms. If this is not the case, then these algorithms may be more efficient than is indicated by Table 1. For Buckley, the function and gradients calls are separated.

As is usually the case in using a program written by someone else, one is never absolutely certain that one has set up the subroutine correctly or that a good or best set of the algorithm parameters has been fed into the program. We tried to take the best set of parameters. In addition, when a program performed unreliably, a different initial point or scaling was used to be certain that with a given function and algorithm a minimum could be obtained under different circumstances.

10. Summary. The tests validate the theoretical results of this paper that an algorithm based upon the SR1 update should perform reliably and efficiently in the presence of box constraints and errors.

For the unconstrained problems tested, with and without errors, the small differences between BCR1(2,1) and Fletcher can be understood by the fact that BCR1(2,1) may use n additional function and gradient evaluations in terminating. The results in Table 1 substantiate the claim that BCR1 is efficient in the presence of error. The results presented for constrained problems do not allow for comparison with an existing, efficient, reliable algorithm. However, BCR1 was reliable on box constrained problems and we feel we can imply from the comparisons on unconstrained problems that it is also efficient.

Moreover, the tests indicate that all the points in the motivation were well taken. In particular, it does seem important to use all the information available to tell when to drop a constraint, and to be able to pick up the true character of a function near a saddle. Moreover, the BCR1 procedure is simple, there is no bookkeeping and no matrix projections and no explicit use of multipliers for the box constraints.

As designed, the procedure handles only box constraints directly. In the circuit design work, more general constraints are treated using penalty functions. It would seem that one could use a program such as this as an inner loop in a penalty–multiplier procedure for the minimization of a general function subject to general constraints.

One final comment. Currently, the algorithm identifies when noise is controlling the procedure and terminates when it makes such an identification. It does this even though the direction that it has is good so that termination would not have happened if a better inital step in the line search had been taken. We are working on this aspect.

References

[1] Cullum, Jane and Brayton, R.K., (1976) Some Remarks on the Symmetric Rank One Update, IBM Research Center Report 6157, Yorktown Heights, New York.

[2] Broyden, C.G. (1967) Quasi-Newton Methods and their Application to Function Minimization, Math. Comp. 21, 368-381.

[3] Fletcher, Roger (1976) Methods for Solving Nonlinearly Constrained Optimization Problems, University of Dundee, Numerical Analysis Report 16.

[4] Powell, M.J.D. (1975) A View of Unconstrained Optimization, C.S.S. 14, AERE, Harwell, England.

[5] Gill, P.E. and Murray, W. (1974) Numerical Methods for Constrained Optimization Academic Press, London and New York.

[6] Mangasarian, O.L., (1972) Dual, Feasible Direction Algorithms, Techniques of Optimization, A.V. Balakrishnan, ed. Academic Press, New York, 67-88.

[7] Fletcher, R. (1973) An Exact Penalty Function for Nonlinear Programming with Inequalities, Math Prog. 5, 129-150.

[8] McCormick, Garth (1969), Antizig-zagging by Bending, Mgnt. Sc. 15 (5), 315-320.

[9] Hestenes, Magnus (1966), Calculus of Variations and Optimal Control Theory, John Wiley and Sons, Inc., New York.

[10] Brayton, R.K., Hachtel, G., and Gustavson, F. (1971) The Sparse Tableau Approach to Network Analysis and Design, IEEE Transactions on Circuit Theory, CT-18 (1), 101-113.

[11] Davidon, William C. (1975) Optimally Conditioned Optimization Algorithms Without Line Searches, Math. Prog. 9, 1-30.

[12] Fiacco, A.V. and McCormick, G.P. (1968) Nonlinear Programming, Sequential Unconstrained Minimization Techniques, John Wiley and Sons, Inc., New York, 170-172.

[13] Murtagh, B.A. and Sargent, R.W.H. (1969) A Constrained Minimization Method with Quadratic Convergence, Optimization, ed. R. Fletcher, 215-246.

[14] Murtagh, B.A. and Sargent, R.W.H. (1970), Computational Experience with Quadratically Convergent Minimization Methods, Computer Journal, 13, 185-194.

[15] Powell, M.J.D. (1970), Rank One Method for Constrained Optimization, Nonlinear and Integer Programming, ed. J. Abadie, North-Holland, Amsterdam.

[16] Powell, M.J.D. (1976), Quadratic Termination Properties of Davidon's New Variable Metric Algorithm, C.S.S. 33, AERE, Harwell, England.

[17] Goldfarb, D. (1969), Extensions of Davidon's Variable Metric Algorithm to Maximization Under Linear Inequality and Equality Constraints, SIAM J. Appl. Math. 17, 739-764.

[18] Gill, P. E. and Murray, W. (1974), Newton-type Methods for Linearly Constrained Optimization, Numerical Methods for Constrained Optimization, ed. P.E. Gill and W. Murray, Academic Press, London and New York, .

[19] Canon, M.D., Cullum, C.D. and Polak, E. (1970), Theory of Optimal Control and Mathematical Programming, McGraw-Hill, New York.

[20] Fletcher, R. and Jackson, M. P., (1974) Minimization of a Quadratic

Function of Many Variables Subject only to Lower and Upper Bounds, J. Inst. Math Applic. 14, 159-174.

[21] Murray, W. (1969), An Algorithm for Constrained Minimization, Optimization, ed. R. Fletcher, Academic Press, London and New York, 247-258.

[22] Powell, M.J.D. (1971) On the Convergence of the Variable Metric Algorithm, J. Inst. Math. Applics. 7, 21-36.

[23] Bard, Y. (1968), On a Numerical Instability of Davidon-Type Methods, Math. Comp. 22, 665-666.

[24] Zoutendijk, G. (1960), Methods of Feasible Directions, Elsevier, Amsterdam.

[25] Himmelblau, D. M., (1972) Applied Nonlinear Programming, McGraw-Hill, New York.

[26] Dixon, L. C. W., (1971) Variable Metric Algorithms: Necessary and Sufficient Conditions for Identical Behavior on Non–quadratic Functions, Numerical Optimization Centre: Technical Report No. 6.

[27] Harwell Subroutine Library (1974), ed. M. J. Hopper, Theoretical Physics Division, Atomic Energy Research Establishment, Harwell, England.

[28] Fletcher, R. (1970), An Efficient, Globally Convergent Algorithm for Unconstrained and Linearly Constrained Optimization Problems, A.E.R.E. Harwell Report T.P. 431.

[29] Buckley, A. (1973) An Alternative Implementation of Goldfarb's Minimization Algorithm, A.E.R.E. Harwell Report T.P. 544.

[30] Fletcher, R. (1970) A New Approach to Variable Metric Algorithms, The Computer Journal, 13, 317-322.

[31] Powell, M. J. D. (1964), An Efficient Method of Finding the Minimum of a Function of Several Variables Without Calculating Derivatives, Computer J. 7, 155-162.

TABLE 1

FCN. NAME	ERROR (REL.,ABS.)	METHOD	FCNS	GRDS	FINAL VALUE	WORK	COMMENTS
FB3	(0,0)	BCR1(1,1)	33	8	-53.5985294	41	(1)
FB3	(0,0)	BCR1(1,4)	30	9	-53.5985294	39	(1)
FB3	(0,0)	BCR1(2,1)	22	11	-53.5985294	33	(1)
FB3	(0,0)	BCR1(2,4)	25	10	-53.5985294	35	(1)
FB3	(0,0)	BUCKLEY	16	15	-8.517287	-	(5),(8)
FB3	(0,0)	CFLETCHER	92	92	6E-4	-	(3),(9)
FB3	(1E-5,1E-6)	BCR1(1,1)	38	11	-53.598526	49	(2)
FB3	(1E-5,1E-6)	BCR1(1,4)	46	12	-53.5985244	58	(2)
FB3	(1E-5,1E-6)	BCR1(2,1)	33	15	-53.5985254	49	(2)
FB3	(1E-5,1E-6)	BCR1(2,4)	27	12	-53.5985293	39	(2)
FB3	(1E-5,1E-6)	BUCKLEY	42	18	-8.5173	-	(5)
FB3	(1E-3,1E-4)	BCR1(1,1)	25	9	1E-3	34	(3)
FB3	(1E-3,1E-4)	BCR1(1,4)	26	10	9.9E-4	36	(3)
FB3	(1E-3,1E-4)	BCR1(2,1)	18	9	1E-3	27	(3)
FB3	(1E-3,1E-4)	BCR1(2,4)	28	11	9.9E-4	39	(3)
FB3	(1E-3,1E-4)	BUCKLEY	120	14	1E-3	134	(4),(3)
FB6	(0,0)	BCR1(1,1)	121	30	-275.49644	151	(1)
FB6	(0,0)	BCR1(1,4)	118	26	-402.311332	144	(1)
FB6	(0,0)	BCR1(2,1)	34	14	-275.49644	48	(1)
FB6	(0,0)	BCR1(2,4)	46	24	-346.091	70	(1)
FB6	(0,0)	BUCKLEY	46	37	-12.402465	-	(5),(8)
FB6	(0,0)	CFLETCHER	480	480	6.0E-6	-	(4),(3)
FB6	(1E-5,1E-6)	BCR1(1,1)	42	12	5E-6	-	(3)
FB6	(1E-5,1E-6)	BCR1(1,4)	36	12	6.3E-7	-	(3)
FB6	(1E-5,1E-6)	BCR1(2,1)	44	17	-275.49644	61	(2)
FB6	(1E-5,1E-6)	BCR1(2,4)	47	23	-346.091	70	(2)
FB6	(1E-5,1E-6)	BUCKLEY	115	24	-12.4021379	-	(5),(7)
FB6	(1E-3,1E-4)	BCR1(1,1)	34	10	6E-5	44	(3)
FB6	(1E-3,1E-4)	BCR1(1,4)	31	10	3E-5	41	(3)
FB6	(1E-3,1E-4)	BCR1(2,1)	34	14	1.2E-4	48	(3)
FB6	(1E-3,1E-4)	BCR1(2,4)	27	14	-1.3E-5	41	(3)
FB6	(1E-3,1E-4)	BUCKLEY	120	13	2.066	-	(4),(5)
FRECP	(0,0)	BCR1(1,1)	60	21	16.5045855	81	(1)
FRECP	(0,0)	BCR1(1,4)	35	12	16.5045855	47	(1)
FRECP	(0,0)	BCR1(2,1)	27	14	16.5045862	41	(1)
FRECP	(0,0)	BCR1(2,4)	38	20	16.5045855	58	(1)
FRECP	(0,0)	BUCKLEY	47	30	16.5045855	77	(1)
FRECP	(0,0)	CFLETCHER	19	19	16.5045776	38	(1)
FRECP	(1E-5,1E-6)	BCR1(1,1)	56	18	16.5045908	74	(2)
FRECP	(1E-5,1E-6)	BCR1(1,4)	43	13	16.5045855	56	(2)
FRECP	(1E-5,1E-6)	BCR1(2,1)	27	14	16.504586	41	(2)
FRECP	(1E-5,1E-6)	BCR1(2,4)	38	20	16.504608	58	(2)
FRECP	(1E-5,1E-6)	BUCKLEY	120	30	16.5045856	150	(4),(2)
FRECP	(1E-2,1E-3)	BCR1(1,1)	31	10	16.6575	41	(2)
FRECP	(1E-2,1E-3)	BCR1(1,4)	31	9	16.51107	40	(2)
FRECP	(1E-2,1E-3)	BCR1(2,1)	24	10	16.736	34	(2)
FRECP	(1E-2,1E-3)	BCR1(2,4)	31	11	16.556	42	(2)
FRECP	(1E-2,1E-3)	BUCKLEY	120	12	17.15	-	(5),(4)

FHOLZ	(0,0)	BCR1(1,1)	112	34	8E-14	146	(1)
FHOLZ	(0,0)	BCR1(1,4)	104	34	4.7E-13	138	(1)
FHOLZ	(0,0)	BCR1(2,1)	69	44	7E-13	113	(1)
FHOLZ	(0,0)	BCR1(2,4)	63	39	1.6E-13	102	(1)
FHOLZ	(0,0)	BUCKLEY	12	8	3.76	-	(6)
FHOLZ	(0,0)	CFLETCHER	181	181	4.171	-	(4)
FHOLZ	(1E-5,1E-6)	BCR1(1,1)	95	29	6.4E-8	124	(2)
FHOLZ	(1E-5,1E-6)	BCR1(1,4)	130	35	4E-7	165	(2)
FHOLZ	(1E-5,1E-6)	BCR1(2,1)	70	39	8E-9	109	(2)
FHOLZ	(1E-5,1E-6)	BCR1(2,4)	34	16	5.6E-3	-	(5)
FHOLZ	(1E-5,1E-6)	BUCKLEY	12	8	3.76	-	(6)
FHOLZ	(1E-2,1E-3)	BCR1(1,1)	38	11	4.94	-	(5)
FHOLZ	(1E-2,1E-3)	BCR1(1,4)	43	11	5.53	-	(5)
FHOLZ	(1E-2,1E-3)	BCR1(2,1)	44	20	5E-3	64	(2)
FHOLZ	(1E-2,1E-3)	BCR1(2,4)	25	9	5.6	-	(5)
FHOLZ	(1E-2,1E-3)	BUCKLEY	120	13	7E-4	133	(4),(2)
FEASY	(0,0)	BCR1(1,1)	11	5	-11.7182934	16	(1)
FEASY	(0,0)	BCR1(2,1)	9	6	-11.7182934	15	(1)
FEASY	(0,0)	FLETCHER	14	14	-11.7182934	28	(1)
FEASY	(0,0)	POW.NO-DRV.	117		-11.7182934	117	(1)
FEASY	(1E-5,1E-6)	BCR1(1,1)	16	5	-11.7182932	21	(2)
FEASY	(1E-5,1E-6)	BCR1(2,1)	16	12	-11.7182929	28	(2)
FEASY	(1E-5,1E-6)	FLETCHER	16	16	-11.7182933	32	(2)
FEASY	(1E-5,1E-6)	POW.NO-DRV	114		-11.7182496	114	(2)
FEASY	(1E-2,1E-3)	BCR1(1,1)	24	10	-11.7182679	34	(2)
FEASY	(1E-2,1E-3)	BCR1(2,1)	20	10	-11.714795	30	(2)
FEASY	(1E-2,1E-3)	FLETCHER	13	13	-11.714877	26	(2)
FEASY	(1E-2,1E-3)	POW.NO-DRV	177		-2.52	-	(5)
FPOWL	(0,0)	BCR1(1,1)	87	29	6.4E-14	116	(1)
FPOWL	(0,0)	BCR1(2,1)	53	45	5E-17	98	(1)
FPOWL	(0,0)	FLETCHER	48	48	3E-11	96	(1)
FPOWL	(0,0)	POW.NO-DRV	290		3E-11	290	(1)
FPOWL	(1E-5,1E-6)	BCR1(1,1)	90	25	2E-7	115	(2)
FPOWL	(1E-5,1E-6)	BCR1(2,1)	35	24	1.3E-6	59	(2)
FPOWL	(1E-5,1E-6)	FLETCHER	38	38	1.3E-7	76	(2)
FPOWL	(1E-5,1E-6)	POW.NO-DRV	309		7.6E-7	309	(2)
FPOWL	(1E-2,1E-3)	BCR1(1,1)	44	12	1E-3	56	(2)
FPOWL	(1E-2,1E-3)	BCR1(2,1)	35	22	7.6E-5	57	(2)
FPOWL	(1E-2,1E-3)	FLETCHER	19	19	3.5E-3	38	(2)
FPOWL	(1E-2,1E-3)	POW.NO-DRV	147		2.6583	-	(5)
FWOOD	(0,0)	BCR1(1,1)	154	44	3E-7	198	(1)
FWOOD	(0,0)	BCR1(2,1)	109	51	2.8E-10	160	(1)
FWOOD	(0,0)	FLETCHER	22	22	7.87	-	(3)
FWOOD	(0,0)	POW.NO-DRV	148		6E-10	148	(1)
FWOOD	(1E-7,1E-8)	BCR1(1,1)	163	48	6.1E-9	211	(2)
FWOOD	(1E-7,1E-8)	BCR1(2,1)	126	53	3.5E-9	179	(2)
FWOOD	(1E-7,1E-8)	FLETCHER	22	22	7.87	-	(3)
FWOOD	(1E-7,1E-8)	POW.NO-DRV	157		6.2E-9	157	(2)

FROSE	(0,0)	BCR1(1,1)	68	21	5.6E-10	89	(1)
FROSE	(0,0)	BCR1(2,1)	61	34	2.5E-11	95	(1)
FROSE	(0,0)	FLETCHER	44	44	1.2E-10	88	(1)
FROSE	(0,0)	POW.NO-DRV	19		4.5E-5	-	(5)
FROSE	(1E-7,1E-8)	BCR1(1,1)	68	21	9.8E-11	89	(2)
FROSE	(1E-7,1E-8)	BCR1(2,1)	62	30	2.7E-10	92	(2)
FROSE	(1E-7,1E-8)	FLETCHER	44	44	1.8E-10	88	(2)
FROSE	(1E-7,1E-8)	POW.NO-DRV	27		4.5E-5	-	(5)
FROSE	(1E-2,1E-5)	BCR1(1,1)	100	32	3.5E-7	132	(2)
FROSE	(1E-2,1E-5)	BCR1(2,1)	66	34	6.1E-7	100	(2)
FROSE	(1E-2,1E-5)	FLETCHER	45	45	1.3E-7	90	(2)
FROSE	(1E-2,1E-5)	POW.NO-DRV	46		2.6E-3	-	(5)

COMMENTS
----

(1) LOCAL MINIMUM
(2) LOCAL MINIMUM TO WITHIN ERROR
(3) STUCK NEAR SADDLE POINT
(4) EXCEEDED MAXIMUM NUMBER OF FUNCTIONS ALLOWED
(5) NOT A LOCAL MINIMUM
(6) ABORTED BY GOING OUT OF BOUNDS
(7) ABORTED BECAUSE OF TOO MANY DIVIDE CHECKS
(8) WENT TO MINIMUM WHEN STARTED AT DIFFERENT POINT
(9) WENT TO MINIMUM WHEN GIVEN DIFFERENT SCALE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 **REPORT DOCUMENTATION PAGE** | READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT ~~NUMBER~~ 18 AFOSR-TR- 77-0904	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) 6 AN ALGORITHM FOR MINIMIZING A DIFFERENTIABLE FUNCTION SUBJECT TO BOX CONSTRAINTS AND ERRORS.		5. ~~TYPE OF REPORT & PERIOD COVERED~~ 9 Interim / rept.
		6. ~~PERFORMING ORG. REPORT NUMBER~~
7. AUTHOR(s) 10 R. K./Brayton and Jane/Cullum		8. CONTRACT OR GRANT NUMBER(s) 15 F44620-76-C-0022
9. PERFORMING ORGANIZATION NAME AND ADDRESS IBM Thomas J. Watson Research Center Yorktown Heights, New York 10598 11 10 Mar 77		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A4 16
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332 12 57p.		12. REPORT DATE 3/10/77 17
		13. NUMBER OF PAGES 48 A4
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) 14 RC-6429		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

constrained optimization, optimization with constraints, box constraints, bound variables, variable metric, quasi-Newton, optimization with errors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We consider the problem of minimizing a differentiable function of n parameters with the parameters restricted to a box in n-space. We describe a variable metric algorithm designed for this problem. This algorithm differs considerably from others proposed. It uses the symmetric rank one update formula, so that no matrix projections at the boundary of the box are necessary; and consequently, a Hessian approximation on the subspace spanned by the directions previously searched is available at each iteration to use in determining which constraints to drop. Moreover, the algorithm is designed to handle problems where the function and its gradient are evaluated with error. The difficulties associated with the

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

349250

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

20 Abstract

symmetric rank one update formula and used to argue against its implementation have seldom been encountered in numerical experience to date and are circumvented, using procedures described in Cullum and Brayton [1]. Numerical examples are presented which show (1) The algorithm presented compares favorably with Fletcher's excellent unconstrained minimization program available from Harwell and (2) On box-constrained problems, the algorithm is reliable and efficient. Problems with and without errors in the function and gradient values were considered.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)